Portals Coding Standards

FSA Portals, Release 1

8/6/2004

Author:	Aimee D. Byrd
Last Modified By:	Aimee D. Byrd
Last Updated:	1/30/2002 12:22 PM
Version:	1.1

Change Record

Date	Author	Version	Change Reference
8/6/2004	Aimee D. Byrd	1.0	Creation of document

1. Introduction

1.1 Purpose

This coding standard document will ensure that programs developed under these guidelines will be easier to read, test, debug, and maintain by various programmers.

1.2 Scope

This document covers coding standards that will be implemented for the various programming languages that will be used in the development of the FSA Portals. These languages include:

- HTML
- Java/Javascript
- CSS
- JSP/Struts

The FSA Portals will comply with Section 508, the Electronic and Information Technology Accessibility Standards. These standards will also be covered in this document.

1.3 Audience

This document is aimed at an audience of technical architects, designers and developers who will be developing the FSA Portals. It is assumed that the audience has a basic understanding of the aforementioned languages.

2. General

2.1 Includes

Include files will be used to improve readability and code reuse. Cascading Style Sheets, tag libraries, and Javascript will each be separately included in a JSP, along with the navigation components each JSP uses. Each navigation component will be in a separate file, so they may be included in a JSP if needed.

Example:

```
<%@ include file="includes/taglib.inc" %>

<%@include file="includes/home.js" %>

<%@include file="styles/students.css" %>

<%@include file="includes/toprightnav.inc" %>
```

2.2 File Names

File names should be in lowercase and as descriptive as possible without going over 32 characters in length. Include files should have the extension of .inc unless the include file is a form (.jsp), a .css, or a .js file. All other files should have their appropriate extension name.

2.3 Image Names

Image names should be in lowercase and as descriptive as possible without going over 32 characters in length. Images can be of any type compatible with Netscape or Internet Explorer.

Images will be in English and Spanish versions, if applicable. The Spanish version of an image will have the exact name as the English version but will include _es at the end of the name.

Example:

```
English:   thinking.college.gif
Spanish:  thinking.college_es.gif
```

2.4 Variables

2.4.1 Declarations

One declaration per line is recommended since it encourages commenting.

Example:

```
int intLevel;   // indentation level
int intSize;  // size of table
```

The example above uses one space between the type and the identifier. Another acceptable alternative is to space out and align the code.

Example:

```
int    intLevel;        // indentation level
int    intSize;         // size of table
```

2.4.2 Names

Variable names should begin with three lowercase characters indicating the variable type.

A list of some data types and their suggested shorthand are in the following table:

Type	Shorthand
Integer	int
Long	lng
Double	dbl
Float	flt
String	str
Character	chr
Array	arr
Date	dte
Boolean	bln

The remainder of the variable name should begin with an uppercase character and be as descriptive as possible. The first letter of any internal words should be capitalized.

Example:

```
intAge, strFirstName, dteBirthday
```

One-character variable names should only be used for temporary variables. Common temporary variables are i, j, k, m, and n for integers and c, d, and e for characters.

Example:

```
for (i = 0; i < intAge; ++ i)

{
   ...
}
```

2.5 Indentation and Braces

Indentation can make or break the readability of code. Indentation should be three spaces and used when applicable.

An opening brace in a statement should line up with its closing brace. See below for an example of how indentation and braces should be handled in the code.

Example:

```
if (intAge < 14)
{
  window.alert(“You must be over thirteen years old to register.”);
}
```

2.6 Whitespace

When used properly, whitespace makes code more readable and easy to follow. However, whitespace can be overused and should be limited. It is recommended that no more than two lines of blank space should be used to separate sections of code. Comments should be the main method of separating one section of code from another.

It is suggested that one space is used in the following places:

- Between operators.
- After semicolons in for-loops.
- Before and after the assignment operator.

2.7 Line Lengths and Breaks

Lines longer than 80 characters should be avoided, since they are not handled well by many terminals and tools.

Break expressions that will not fit on a single line according to these general principles:

- Break after a comma.
- Break before an operator.
- Prefer higher-level breaks to lower-level breaks.
- Align the new line with the beginning of the expression at the same level on the previous line.

2.8 Comments

Comments are an integral part of code documentation. A developer spends much less time deciphering existing code when good comments are provided.

Each file should begin with comments. Beginning comments serve as a file header and provide information on the file name, the date the file was created, a description of the file, and any copyright notices.

Example:

```
/*
* name:
* date created:
* description:
* copyright:
*/
```

Comments should also be provided throughout the code to make the code more readable and easy to understand.

3. Resource.properties

3.1 Usage

The Resource.properties file will be used for internationalization values only. Any images or text that will appear in the FSA Portals in English and other languages should be given key names and the actual value put in the Resource.properties file for each language supported by the Portals.

HTML tags may be used but should be limited in the Resource.properties file.

3.2 Key Structure

There will be one Resource.properties file for both the Students and Financial Partners Portals. Thus, the key names should be structured for easy readability and maintainability.

The Resource.properties file should be in sections in the following order:

- FP Keys
  - General
  - Page
    - Any order
- Students Keys
  - General
  - Page
    - Any order

Example:

```
# FP Keys##########################

# General

fp.home.lnk=Home

# Home Page – Default page of the FSA Financial Partners Portal

fp.home.title.txt=FSA Financial Partners Portal – Home



# Students Keys#####################

# General

st.home.lnk=Home

# Home Page – Default page of the FSA Students Portal

st.home.title.txt=FSA Students Portal - Home
```

Each section and sub-section should include comments describing the section. No standard is set for commenting in the Resource.properties file. However, be consistent when commenting.

3.3 Names

Key names should be in the form:

<portal type>.<description>.<area (optional)>.<index (optional)>.<type>

where **<portal type>** = fp or st (for Financial Partners or Students)

and **<description>** = description of value or page name

and **<area>** = occurrence on page (paragraph 1 or beginning of sentence)

and **<index>** = the number of the occurrence (if there are 2 or 3 images, may want to indicate which image referencing)

and **<type>** = element type (image, text).

Example:

```
st.home.title.text
fp.faqs.questionheader.1.text
```

3.3.1 Shorthand Suggestions for Type

Type	Shorthand
Image	img
Text	txt
Link	lnk
Label	lbl
Button	btn

4. HTML

4.1 Tags

All uppercase is recommended but not required for tags such as: <HTML/>, <HEAD/>, <BODY/>, <FORM/>, and <TABLE/>. However, the usage or non-usage of all uppercase tags should be consistent throughout the code.

4.2 Colors

The standard for defining colors is to use a color's hexadecimal notation.

Example:

```
<font color=#000000>Home</f>
```

4.3 Quotations

Quotations are suggested but not required when referencing a file or website. However, the usage or non-usage of quotations should be consistent throughout the code.

Example:

```
<a href = “http://www.ed.gov” target=”_blank”>Department of Education</a> or <a
href = “repayment/repayingloans.html”>Repaying Loans</a>
```

5. Java

5.1 Constants

The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_").

Example:

```
public static final int DEFAULT_COLOR = Color.black;
```

5.2 Classes

Class names should be as simple yet descriptive as possible. The use of whole words rather than acronyms or abbreviations is recommended.

5.3 ITA Java Coding Standards

Please reference Task Order #16, Deliverable #16.1.3 – Java Coding Standards for more Java coding standards and techniques. Due to the limited amount of Java code in the FSA Portals, the Java Coding Standards document may be viewed as suggested and not required standards.

6. Section 508

6.1 Guidelines

- A text equivalent for every non-text element must be provided. (via “alt”, “longdesc”, or in element content). Examples of non-text elements are images, multimedia objects, logos, photographs, artwork, and applets.

- Equivalent alternatives for any multimedia presentation must be synchronized with the presentation.

- All information conveyed with color must be available without color.

- Documents must be readable without requiring an associated style sheet.

- Redundant text links must be provided for each active region of a server-side image map.

- Client-side image maps must be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

- Row and column headers must be identified for data tables.

- Frames must be titled with text that facilitates frame identification and navigation.

- The screen must not flicker with a frequency greater than 2 Hz and lower than 55 Hz.

- A text-only page, with equivalent information or functionality must be provided when 508 compliance cannot be accomplished any other way and must be updated accordingly. PDF files may need a text-only alternative because in certain cases software readers do not recognize them accurately. You need to test your PDF files using different readers to decide if text-only versions are needed.

- Electronic forms must allow people using assistive technology to access the information, field elements and functionality required for completion and submission of the form.
- A user must be permitted to skip repetitive navigation links.
- When a timed response is required, the user must be alerted and given sufficient time to indicate more time is required.
- When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script must be identified with functional text that can be read by adaptive technology.

6.2 Useful Links

- http://www.niehs.nih.gov/websmith/508/home.htm - Quick reference guide that includes an evaluation form and links to assessment tools.
- http://www.sun.com/access/developers/access.quick.ref.html - Provides a quick reference guide to accessibility.
- http://www.macromedia.com/macromedia/accessibility/ - Macromedia provides a free download that assesses a web page's 508 compliance. Must be run on Dreamweaver.
- http://webdesign.about.com/cs/accessibility/index.htm - Collection of links to various web-accessibility sites.